CPSC 101 Team Term Project

Winter 2013

Dr David Casperson

Note This problem statement is incomplete and will likely be revised.

Problem statement

The goal of this project is to write programs to play the two-person variant of the card game called cribbage.

The Game Cribbage is played between two players with a cribbage board and a standard 52-card deck of cards (the jokers are not used). Although there are variants of cribbage for three or four players, for the purposes of this project cribbage is a two-player game. The cribbage board is essentially a mechanical counting device capable of keeping track of two totals that are between 0 and 121 and of displaying the last number added to each total.

The object of the game is to be the first player to score (at least) 121 points.

At any given point in the game one player is the dealer and the other is the *pone* (non-dealer). In any given game being the dealer alternates between the players. In the first game of the sequence the dealer is determined by drawing cards. Thereafter, the loser of the previous game deals first in the next game.

A round During each round of play the dealer starts by dealing six cards face down to herself and her opponent. Each player then picks up their hand and selects two cards to discard face down into the *crib*, which belongs to the dealer. After the pone has discarded, she cuts the remainder of the deck. After the dealer has discarded and the pone has cut the deck, the dealer turns the top card of the face up. If this card is a Jack, the dealer immediately scores two points.[1]

After discarding and cutting, the players take turns exposing their cards as described below under "pegging". Assuming that neither player wins during the pegging of the hands, the hands are then counted for points. The pone counts her hand first. The dealer then counts her hand. Finally, the dealer then counts the crib. The order is important because either player may win when it is their turn to count. There are no draws in cribbage.

Pegging During pegging, the players take turns exposing their cards with the non-dealer starting. As each player exposes a card, she announces the total count and possibly the number of points scored. Thus a player might say "twenty" or "twenty for three". For the purposes of counting aces count as one; jacks, queens and kings count as ten, and other cards count as their face value.

While pegging, the count never exceeds thirty-one. If a player cannot play a card without exceeding thirty-one, she says "go" instead. Her opponent then plays any cards she legally can (in any order she chooses) until either the count reaches thirty-one, or she also says go.

If the count reaches thirty-one or both players say "go" and at least one player has cards remaining, the count restarts at zero. If both players have cards remaining, the player who did not play a card last plays first. Pegging stops when both players are out of cards.

[1] The dealer can win the game this way.

Scoring The dealer may score when the cut card is exposed if it is a jack. Either player may score when it is their turn to play during the pegging. After the pegging each player scores their hand. When scoring a hand a player uses five cards: the hand (or crib) and the cut card. Scoring is as follows:

The cut card If the cut card is a jack it counts as two points for the dealer.

Pegging

- If a player causes the count to reach exactly fifteen or exactly thirty-one, she scores two points. If a player plays the last card of a counting sequence and the count does not reach thirty-one, she score one point.
- If a pair of cards of the same rank are played sequentially (without an intervening restart of counting), the player who plays the second card scores two.
- If a three cards of the same rank are played sequentially (without an intervening restart of counting), the player who plays the third card scores six.
- If a four cards of the same rank are played sequentially (without an intervening restart of counting), the player who plays the fourth card scores twelve.
- If three or more cards played in order (without an intervening restart of counting) form a sequence (possibly out of order) the player who plays the third or later scores the length of the sequence.

Counting a hand

- Each subset of a hand whose count is exactly fifteen counts for two points.
- Each pair counts for two.
- (thus) each triple counts for six, and
- each quadruple counts for twelve.
- Each maximal sequence of three or more cards counts for the length of the sequence.
- A jack of the same suit as the cut card counts for one.
- If the cut card and all of the cards in a hand or the crib are of the same suit, it counts for five. Otherwise if all of the cards in a hand are of the same suit, it counts for four.[2]

[2]This does not work for the crib.

```
    Your cards:  AS 2H 6D 10S JS QS

.. ..... ..O.. ..... .O... ..... ..... ..... ..... ..... ..... ..... .....
.. ..... .O... ..O.. ..... ..... ..... ..... ..... ..... ..... ..... .....

Cut card:  ??

Computer cards:  2S 3H 7D JC QC KC
```

Figure 1: ASCII rendition of a cribbage board.

Programming tasks

Your team needs to write two or three separate programs: a "referee", a human player interface, and a computer opponent. You may choose to combine the referee and human player interface if you so choose.

The Computer Opponent The computer opponent is a stand-alone program that reads commands from standard input (`System.in`) and writes responses to standard output (`System.out`).

The computer opponent program is an "artificial intelligence" only. It does not display output or offer a graphical user interface. The actual language of commands and responses will be detailed in a later handout. The computer opponent should make minimal assumptions about the order of the sequence of commands that it receives. The responses should be exactly as described in the later handout. Note that the computer opponent must keep track of the current board position. Each command ends with a semi-colon (`;`) followed by a new-line.[3] Each response ends in a period followed by a newline.

The Referee The referee program minimally allows a person to play cribbage against the computer. It keeps track of whether either player has won the current game; lets the human player quit or restart the game at any time (s)he chooses; and draws (or otherwise displays) the board so that the human player can see the current game situation.

ASCII-graphics such as those shown in Figure 1 are acceptable for displaying the board. However, a Graphical User Interface is preferred.

The exact mechanism that the referee program uses for interacting with the computer opponent will be explained in more detail in a later handout. Generally speaking the referee will send simple commands through an `PrintWriter` to the computer opponent, and the computer opponents response to the commands will appear in an `BufferedReader`.

---

[3] Actually, it is better to do this in a platform independent manner. In JAVA, the appropriate end-of-line sequence is not known at compile-time, as a JAVA `.class`-file is supposed to run on multiple operating systems, which may have differing line-ending conventions. The correct multi-platform way to do this is to use a `java.io.PrintWriter`, and then use `.println(...)` or `.format("... %n", ...)`, possibly followed by a `.flush()` (the documentation indicates that `PrintWriter` will take care of the flush when the programmer uses "`.println`" or "`.printf`". In fact whehter or not this happens is controlled by an argument to the constructor of the `PrintWriter`.)

General comments

The referee program and the computer opponent program make use of similar concepts, so they should make use of common classes and object files in their construction. *Your coding will be graded in part on how much code is shared between the two program;* as one of the goals of good object oriented programming is to create classes and objects that can be used in multiple program.

For the computer opponent program, correctness is far more important than cleverness. The computer opponent program must work correctly, even when used by a referee program other than your own. However, intelligent play by the computer opponent is not necessary, and should not be a priority when completing the project.